SSBRP COMMUNICATION & DATA SYSTEM DEVELOPMENT USING THE UNIFIED MODELING LANGUAGE (UML)

May Windrem † and Lou Picinich ††

NASA Ames Research Center
MS 244-19
Moffett Field, CA 94087, USA
FAX: 415-604-0673, † E-mail: mwindrem@mail.arc.nasa.gov
†† E-mail: lpicinich@mail.arc.nasa.gov

ABSTRACT

The Unified Modeling Language (UML) is the standard method for specifying, visualizing, and documenting the artifacts of an object-oriented system under development. UML is the unification of the object-oriented methods developed by Grady Booch and James Rumbaugh, and of the Use Case Model developed by Ivar Jacobson. This paper discusses the application of UML by the Communications and Data Systems (CDS) team to model the ground control and command of the Space Station Biological Research Project (SSBRP) User Operations Facility (UOF). UML is used to define the context of the system, the logical static structure, the life history of objects, and the interactions among objects.

1. INTRODUCTION

Since the inception of the project, the CDS team has chosen an object-oriented approach for system development for the following reasons:

- to better manage the complexity of the system
- to capitalize on software re-use
- to avoid the ripple effects caused by changes to the requirements.

Before 1996, the CDS team used the Object Modeling Technique (OMT) for its software development. Due to the following two factors, the CDS team switched to UML in 1996:

- UML is the successor of OMT
- UML incorporated Use Cases as a means to capture and specify user requirements

This paper covers the following topics:

- a brief description of UML
- how the CDS team defined the CDS requirements using UML
- how the CDS team is designing CDS using UML

2. UML OVERVIEW

UML is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. It defines the notation and semantics for modeling systems using object-oriented concepts. UML does not define the software development process; it does not describe how to do object-oriented analysis and design. However, its authors recommend a use-case driven, architecture-centric, iterative, and incremental process. CDS has followed this approach.

UML was initially created by Booch (of Booch method), Rumbaugh (co-developer of the OMT method), and Jacobson (of use cases) in the hope of bring together the best of the various OO modeling approaches. In 1997, UML was adopted as a standard by the Object Management Group (OMG), an industry standards body.

3. CDS OVERVIEW

CDS will provide a complete operational environment for the ground-based SSBRP User Operations Facility at NASA Ames Research Center. The functionality of CDS is divided into two categories:

1. core functions:
   - commanding the SSBRP hardware onboard the International Space Station (ISS)
   - monitoring the SSBRP hardware onboard ISS
   - processing the SSBRP telemetry data downlinked to the User Operations Facility (UOF) at NASA Ames Research Center
   - monitoring the SSBRP telemetry data
   - distributing SSBRP telemetry data to the Princpal Investigators
   - commanding and monitoring the Ground Controls hardware
   - processing and monitoring the simulated telemetry data from the Ground Controls hardware
2. auxiliary functions, such as payload planning, logistics, and network monitoring

The core functions are included in the CDS Core System. Due to the greater importance of the core functions, the CDS Core System will be developed first.

All of the CDS functions will operate in four different modes: Operations, Simulation, Training, and Testing. The functions needed to support the Simulation, Training, and Testing modes are subsets of the functions performed in the Operations mode. The operations mode of the CDS Core System will be delivered first because it is the most critical part of the system. Consequently, the CDS team is concentrating on the analysis and design of the Core system.

The context diagram shown in Figure 1 depicts the boundary of the CDS Core System and shows all the external entities, called actors, that will interact with it. The actors can be people, such as operators, or other systems, such as the Payload Data Services System (PDSS) and the Payload Operations Integration Function (POIF). Actors supply information to CDS, retrieve information from CDS, or do both.

4. USE CASES

Use Cases are used to discover and specify the behavior of a system. They specify requirements, major functions, actors, and the interactions of actors with CDS. A Use Case must be initiated by an actor. The use case diagram shown in Figure 2 depicts all the use cases of the CDS Core System in the operations mode. *(Note: Use cases are named from the point of view of the actor, not CDS. For example, PDSS Sends Telemetry to CDS, not CDS receives telemetry from PDSS.)*

CDS uses a template to document a Use Case. An example of a template is shown in Figure 3. Part 1 of the template, Use Case, gives the name of the Use Case. Part 2, Actor, lists the actor or actors that can initiate the use case. Part 3, Description, describes the typical execution of this use case. Part 4, Side Effects, lists any consequences of an execution of the use case that are not readily discernible from the description.

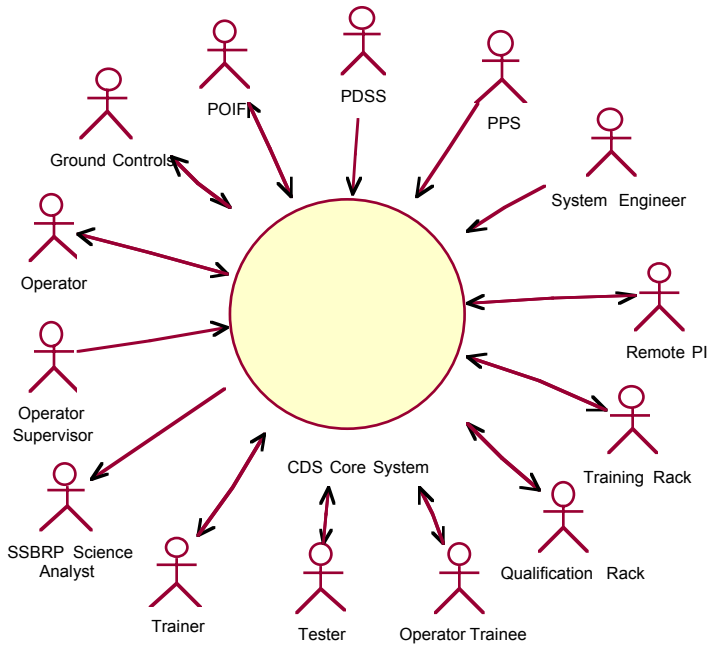

Figure 1. CDS Core System Context Diagram

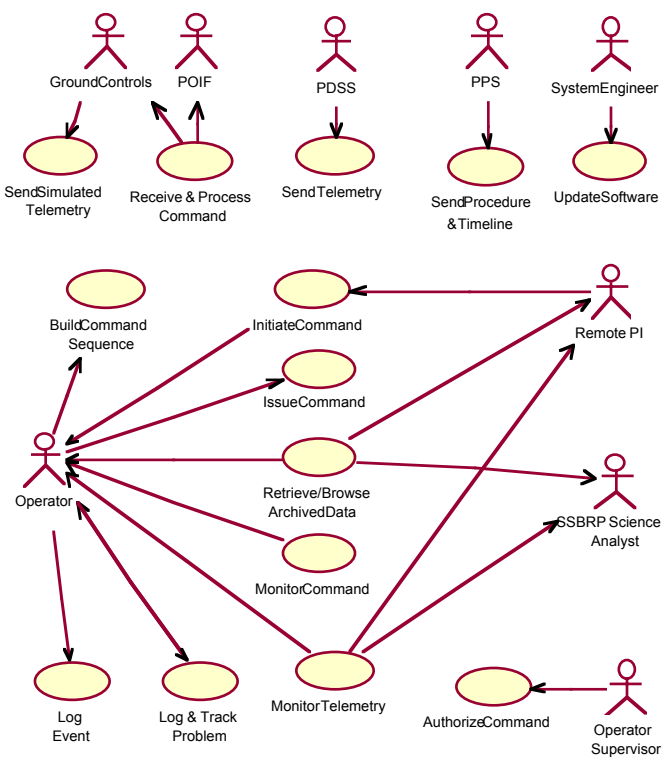

Figure 2. CDS Core System Operations Mode Use Case Diagram

CDS Core System *Operations Mode*

Use Case:
Send Telemetry

Actor:
Payload Data Services System (PDSS)

Description:

PDSS processes and distributes return link core and payload telemetry data downlinked from the ISS. It also provides production data processing, line outage recording, 7 days of online data storage, and 2 years of offline data storage. PDSS is hosted at the Enhanced Huntsville Operations Support Center in MSFC. PDSS sends the processed SSBRP telemetry data to CDS.

Side Effects:

PDSS in its current design does not process system requests to resend stored telemetry data. Therefore, UOF personnel have to log on the Payload Information Management System (PIMS) manually to request the retransmission of data, rather than the retransmission being a part of the communication protocol.

Figure 3. Example of a Use Case Template

5. CLASS DIAGRAM

Class diagrams are tools that the CDS team used to document the problem domain. In a class diagram, the classes are represented by rectangles. Each rectangle is divided into three sections. The name of a class is in the top section. The attributes of a class are in the middle section. The methods, or functions, of a class are in the bottom section. In the analysis phase of CDS, the team identified the objects and classes that comprise the problem domain. The class diagram shown in Figure 4 is a high level view of the classes and their relationships, such as associations, generalization, and aggregation. The attributes and methods are not shown in Figure 4, 5, or 6.

The CDS Project is in the early design stage. Besides the problem domain classes, the team has also defined some implementation classes as shown in Figures 5 and 6.

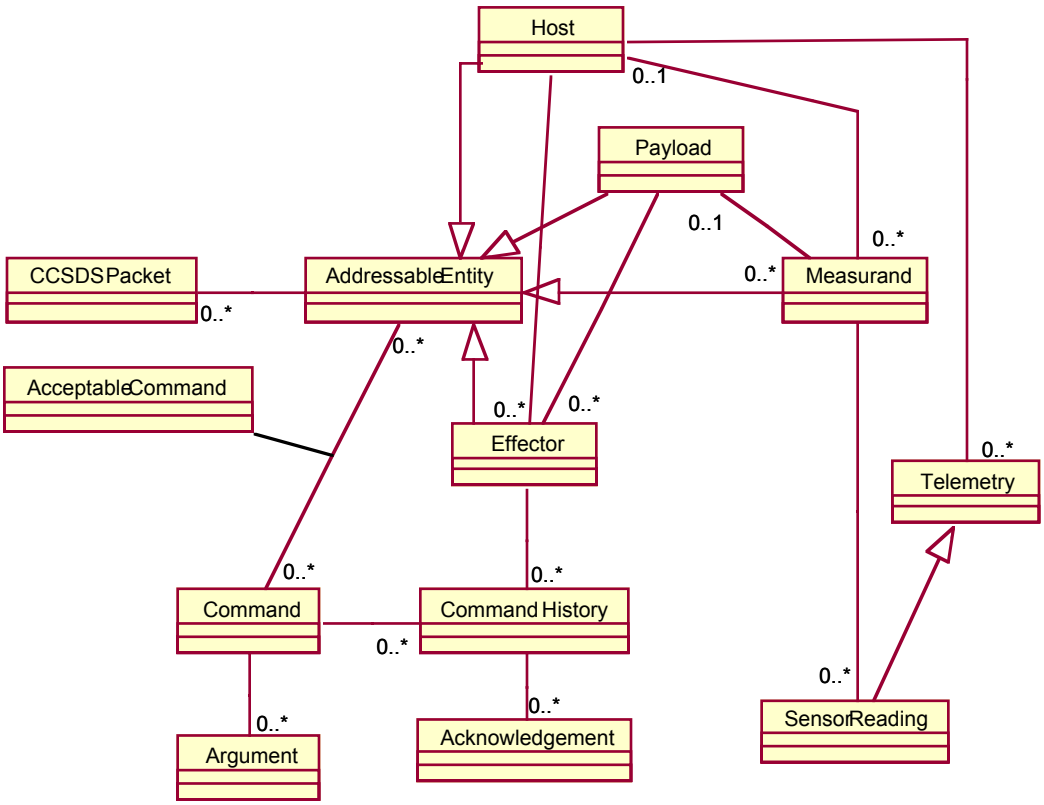

Figure 4. Problem-Domain Class Diagram

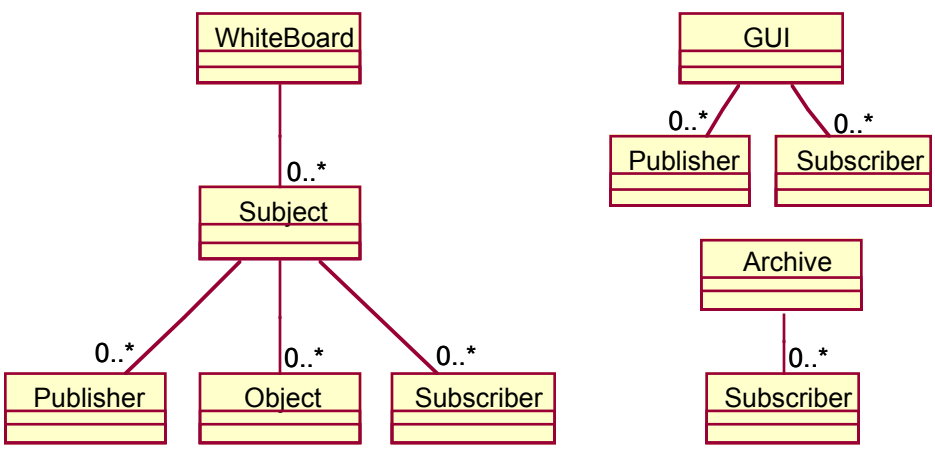

Figure 5. CDS Core Real Time System Class Diagram

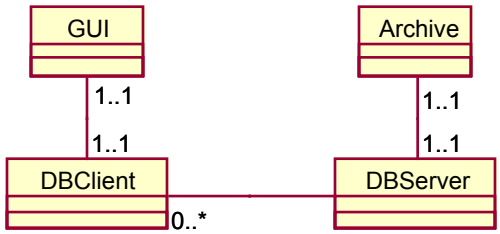

Figure 6. CDS Core Off Line System Class Diagram

6. SEQUENCE DIAGRAM

A Use Case consists of a set of related sequences of interactions between an actor and the system. Each sequence in the Use Case is called a Scenario. In other words, each path through a Use Case is called a Scenario. A sequence diagram shows the objects involved in satisfying the system level requirement, the events and messages that are sent by each object, and the sequence of interactions between objects. This diagram does not show switching, branching, or other forms of decision making. A sequence diagram is generated for each of the Scenarios. It describes how the system will execute a particular Scenario. In a sequence diagram, objects are shown as vertical lines, messages and events are shown as directed lines between objects, external actors are shown as solid circles, and time moves from top of the diagram to the bottom. Figure 7 shows how CDS will execute one Scenario of the PDSS Sends Telemetry Use Case.

Use Case: PDSS Sends Telemetry

Scenario: CDS receives telemetry from PDSS. The formatter decodes the telemetry packets. The publisher publishes the telemetry data to the whiteboard. Archive gets the telemetry data from the whiteboard and stores it in the database. The CDS real-time monitor displays the telemetry data on the screen for the operator.

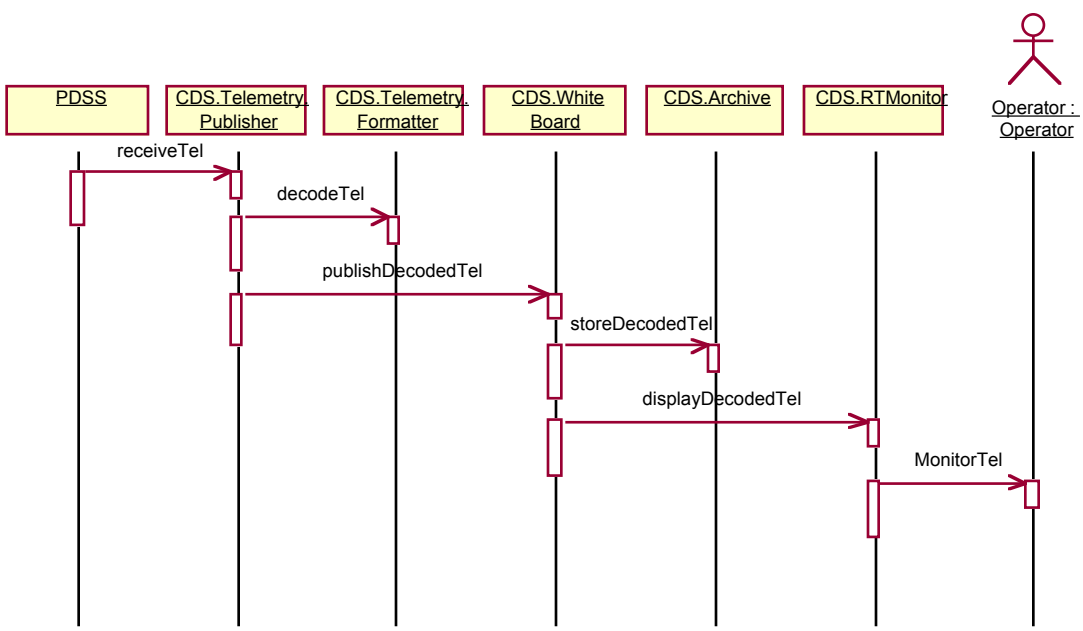

Figure 7. PDSS Send Telemetry Sequence Diagram

7. STATE DIAGRAM

State diagrams depict the life-cycle of an object--the events it experiences, its transitions, and the states it is in between the events. State diagrams show the start state, the transitions to one or more states, and one or more end states. Figure 8 shows the state transitions of a CDS command object. The transitions are shown as arrows, labeled with their events. States are shown in rounded rectangles.

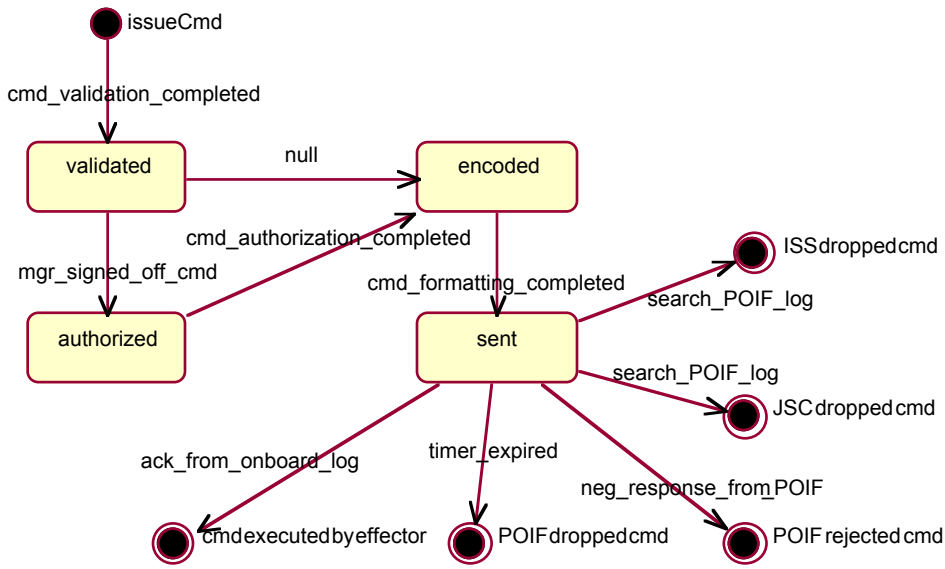

Figure 8. State Transition Diagram for CDS Commands

8. CONCLUSION

In the CDS team's experience, the Unified Modeling Language provides the necessary vocabulary for defining and designing CDS. The following diagrams define CDS:

- The Use Case diagrams describe CDS's interaction with the outside world and define its requirements.
- The Class Diagrams define the problem-domain classes and implementation classes.
- The Sequence Diagrams describe the behavior and interactions of objects.
- The State Diagrams describe the life-histories of objects.

To complete the design phase of the project, the CDS team will perform the following tasks:

- define reports
- design user interfaces
- refine the system architecture
- define interaction diagrams
- refine the implementation class diagrams
- define the database schema.

The CDS team will build the system according to the design described in this paper. The team will keep these diagrams updated as system documentation.

REFERENCES

1. Space Station Biological Research Project (SSBRP) http://pyroeis.arc.nasa.gov
2. Unified Modeling Language (UML), Rational Corporation http://www.rational.com